LA-UR-18-26553
Approved for public release; distribution is unlimited.

Title: What is it: Adjoint Methods in Optimizations?

Author(s): Godinez Vazquez, Humberto C.

Intended for: internal presentation

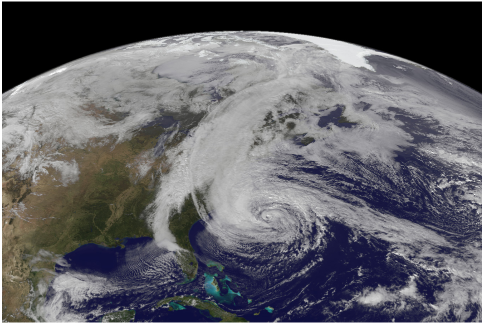
Issued: 2018-07-16

What is it: Adjoint Methods in Optimizations?

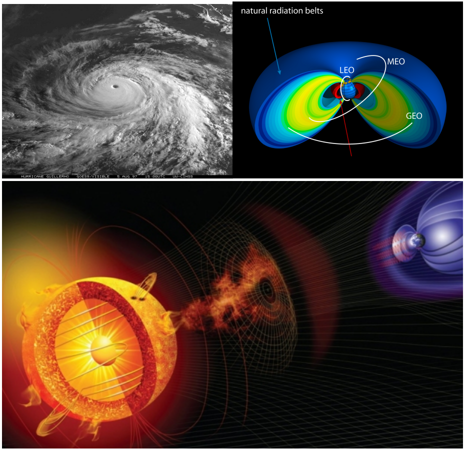
Humberto C. Godinez
Los Alamos National Laboratory

July 16, 2018

Introduction: Who cares?



In 2012 Hurricane Sandy made landfall near NY, causing 53 fatalities and \$50 billion in damage (FEMA P-942) It has been acknowledged that error in model simulation, caused by uncertainties in model inputs, was the main culprit in accurately forecasting Sandy (McNally et al. 2014, Bassill 2014, Cohn 2015).

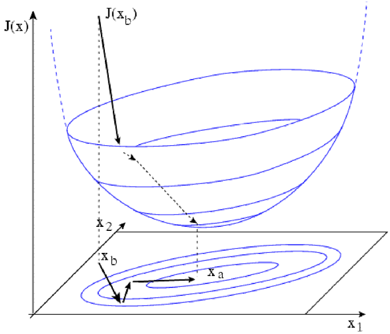


Data assimilation are methods that combine information from a model, observational data, and corresponding error statistics, to provide an estimate of the true state of a system as accurately as possible.

These methodologies are used in a wide range of problems, such as:

- Weather prediction
- Hurricane simulation and forecasting
- Radiation belt simulation
- Solar Physics

Optimization for DA



The main idea is to minimize a cost or penalty function $\mathcal{J}$ which is defined as

$$\mathcal{J}\left(\mathbf{x}_0\right) = \|\mathbf{y}^o - \mathbf{x}_i\|$$

where

$$\mathbf{x}\left(t_i\right) = \mathcal{M}_{t_0 \to t_i}\left(\mathbf{x}\left(t_0\right)\right) \qquad (1)$$

The solution of this minimization problem is performed iteratively with a Newton type technique (steepest decent). The analysis is the minimum of the cost function

$$\mathbf{x}^a = \underset{\mathbf{x}_0 \in \mathbb{R}^n}{\text{argmin}} \mathcal{J}\left(\mathbf{x}\right)$$

Tangent and Adjoint are needed for both methods!!

Adjoint Models for Derivatives

$$\underset{\text{Input}}{\mathbf{x}_0} \longrightarrow \boxed{\mathcal{M}_{t_0 \to t_v}} \longrightarrow \underset{\text{Output}}{\mathbf{x}_v} \qquad \underset{\text{Input}}{\mathbf{x}_0 + \delta\mathbf{x}_0} \longrightarrow \boxed{\mathcal{M}_{t_0 \to t_v}} \longrightarrow \underset{\text{Output}}{\mathbf{x}_v + \delta\mathbf{x}_v}$$

Tangent (first order derivatives) of functions (any input-output relation) can be computed through definition of adjoint variables

Methods for computing derivatives include:

- Deterministic:(Sensitivity Analysis, $\dfrac{d\mathbf{x}_v}{d\mathbf{x}_0} \approx \dfrac{\delta\mathbf{x}_v}{\delta\mathbf{x}_0}$)
  - Form set of Ordinary Differential Equations (ODE) to approximate derivative of model w.r.t. parameters
  - Automatic Differentiation (AD)
- Statistical:(Uncertainty Quantification)
  - Monte-Carlo methods

Tangent Linear Model

Dynamical system

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(t, x; p) \tag{2a}$$

$$x(t_0) = x_0, \quad t_0 \le t \le t_f \tag{2b}$$

where p are the model parameters. Notice $x = x(t; x_0, p)$. For $\delta x_0 \Rightarrow \delta x$, the sensitivity is defined $s(t) = \dfrac{\partial x}{\partial x_0}$. Differentiating (2) wrt y_0 we obtain the Tangent Linear Model (TLM)

$$\frac{\mathrm{d}s}{\mathrm{d}t} = \frac{\partial f}{\partial x}(t, x; p)\, s \tag{3a}$$

$$s(t_0) = e_i \tag{3b}$$

The solution $s(t)$ provides with the forward sensitivity.

Adjoint Model

Some applications require the sensitivity of a scalar response functional $\mathcal{J} = \mathcal{J}\left(x\left(t_f\right)\right)$. A perturbation δx_0 generates $\delta\mathcal{J} = \mathcal{J}\left(x_f + \delta x_f\right) - \mathcal{J}\left(x_f\right)$
To a first order approximation

$$\delta\mathcal{J} = \left\langle \nabla_{x_f}\mathcal{J}\left(x_f\right), \delta x_f \right\rangle = \left\langle \nabla_{x_0}\mathcal{J}\left(x_f\right), \delta x_0 \right\rangle \tag{4}$$

δx_f can be computed through the TLM

$$\begin{aligned} \frac{\mathrm{d}\delta x}{\mathrm{d}t} &= \frac{\partial f}{\partial x}(t, x; p)\,\delta x && (5) \\ \delta x\left(t_0\right) &= \delta x_0 && (6) \end{aligned}$$

Introduce λ, take inner product of (5)-(6), integrate on $[t_0, t_f]$ to obtain

$$\int_{t_0}^{t_f} \left\langle \lambda, \frac{\mathrm{d}\delta x}{\mathrm{d}t} \right\rangle dt = \int_{t_0}^{t_f} \left\langle \lambda, \frac{\partial f}{\partial x}(t, x; p)\,\delta x \right\rangle dt$$

$$\int_{t_0}^{t_f} \left\langle \lambda, \frac{\mathrm{d}\delta x}{\mathrm{d}t} \right\rangle dt = \int_{t_0}^{t_f} \left\langle \left[\frac{\partial f}{\partial x}(t, x; p) \right]^* \lambda, \delta x \right\rangle dt$$

integrating the left side by parts

$$\langle \lambda, \delta x \rangle |_{t_0}^{t_f} = \int_{t_0}^{t_f} \left\langle \frac{\mathrm{d}\lambda}{\mathrm{d}t} + \left[\frac{\partial f}{\partial x}(t, x; p) \right]^* \lambda, \delta x \right\rangle dt \tag{7}$$

define λ as the solution of the First Order Adjoint (FOA) system

$$\frac{\mathrm{d}\lambda}{\mathrm{d}t} = - \left[\frac{\partial f}{\partial x}(t, x; p) \right]^* \lambda \tag{8}$$

$$\lambda(t_f) = \nabla_{x_f} \mathcal{J}(x_f), \quad t_f \geq t \geq t_0 \tag{9}$$

equation (7) reduces to

$$\left\langle \nabla_{x_f} \mathcal{J}(x_f), \delta x_f \right\rangle = \langle \lambda_0, \delta x_0 \rangle = \delta \mathcal{J}$$

To obtain the sensitivity we integrate (8)-(9) backward only once and compute the inner product for any δx_0.

Los Alamos
NATIONAL LABORATORY
EST. 1943

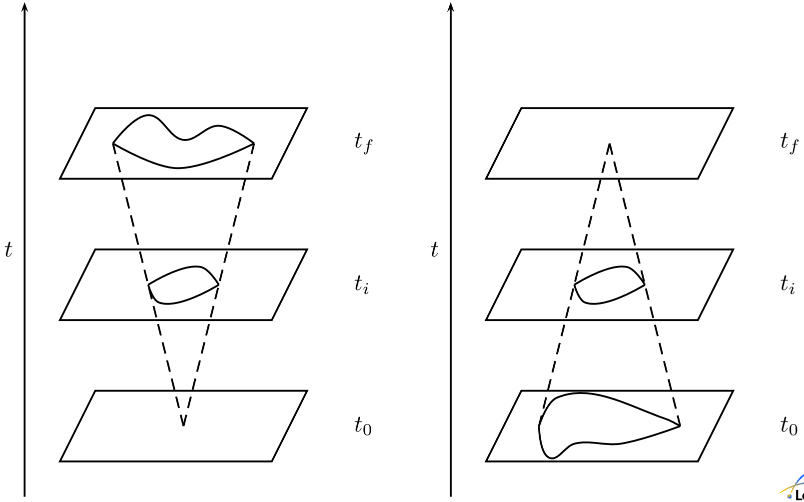
The solution of the FOA equations λ_0 gives the gradient of $\mathcal{J}$ with respect to x_0

$$\delta\mathcal{J} = \langle \nabla_{x_0}\mathcal{J}\left(x_f\right), \delta x_0 \rangle = \langle \lambda_0, \delta x_0 \rangle$$

so we have that

$$\lambda_0 = \nabla_{x_0}\mathcal{J}\left(x_f\right)$$

Graphical representation



Implementation of Tangent and Adjoint Models

Continuous tangent and adjoint

- Does not depend on code
- Has to be discretized and solved separately
- Discretization has to be consistent

Discrete tangent and adjoint (Automatic Differentiation)

- Depends on code
- Consistency
- TAMC, ADIFOR,etc..

Automatic Differentiation

- Computational technique to obtain the tangent linear model or adjoint model of a code that is smooth or differentiable.
- View code as a function with input variables and output variables.
- Use basic rules of differential calculus to obtain tangent code line by line.

Algorithm 1 Forward Model

```
function 𝓜(x,y,z)
    z = sin x + y^2
end function
```

Algorithm 2 Tangent Model

```
function M(x,y,z,δx,δy,δz)
    δz = cos(x)δx + 2yδy
    z = sin x + y^2
end function
```

$$\delta z = \begin{pmatrix} \cos(x) & 2y \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$

$$\begin{pmatrix} \delta x^* \\ \delta y^* \end{pmatrix} = \begin{pmatrix} \cos(x) \\ 2y \end{pmatrix} \delta z^*$$

Algorithm 3 Tangent Model

1: **function** $\mathbf{M}(x,y,z,\delta x,\delta y,\delta z)$
2: $\quad \delta z = \cos(x)\delta x + 2y\delta y$
3: $\quad z = \sin x + y^2$
4: **end function**

Algorithm 4 Adjoint Model

1: **function** $\mathbf{M}^*(x,y,\delta x^*,\delta y^*,\delta z^*)$
2: $\quad \delta x^* = \cos(x)\delta z^* + \delta x^*$
3: $\quad \delta y^* = 2y\delta z^* + \delta y^*$
4: $\quad \delta z^* = 0.0$
5: **end function**

Discrete Tangent

In practice, the system (2) is solved numerically, so a discrete version of the adjoint is needed.
Let

$$\mathbf{x}_{i+1} = \mathcal{M}_i\left(\mathbf{x}_i\right), \quad i = 0, \ldots, N-1 \tag{10}$$

be the discrete time evolution of the system (2) after a time discretization is applied. Let $\mathbf{M}_i$ be the *discrete tangent linear model* of $\mathcal{M}_i$, i.e.

$$\mathbf{M}_i\left(\mathbf{x}_i\right) = \frac{\partial \mathcal{M}_i}{\partial \mathbf{x}_i}\left(\mathbf{x}_i\right) \tag{11}$$

Using (11) the discrete Tangent Linear Model (TLM) is given by

$$\mu_0 = \mathbf{w} \tag{12}$$

$$\mu_{i+1} = \mathbf{M}_i\left(\mathbf{x}_i\right)\mu_i, \quad i = 0, \ldots, N-1, \tag{13}$$

Discrete Adjoint

Let $\mathcal{J} = \mathcal{J}(\mathbf{x}_N)$ be a response functional. Introduce $\delta\mathbf{x}_0 \Rightarrow \delta\mathcal{J}(\mathbf{x}_N)$, as before

$$\delta\mathcal{J} = \langle \nabla_{\mathbf{x}_0}\mathcal{J}(\mathbf{x}_N), \delta\mathbf{x}_0 \rangle$$

We want to compute $\nabla_{\mathbf{x}_0}\mathcal{J}(\mathbf{x}_N)$. Using the chain rule we have

$$\nabla_{\mathbf{x}_0}\mathcal{J}(\mathbf{x}_N) = \nabla_{\mathbf{x}_0}\mathbf{x}_1 \nabla_{\mathbf{x}_1}\mathbf{x}_2 \cdots \nabla_{\mathbf{x}_{N-1}}\mathbf{x}_N \nabla_{\mathbf{x}_N}\mathcal{J}(\mathbf{x}_N)$$

Notice

$$\nabla_{\mathbf{x}_i}\mathbf{x}_{i+1} = \left(\frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{x}_i}\right)^T = \left(\frac{\partial \mathcal{M}_i}{\partial \mathbf{x}_i}(\mathbf{x}_i)\right)^T = \mathbf{M}_i^T(\mathbf{x}_i)$$

where $\mathbf{M}_i^T$ is the discrete adjoint of $\mathbf{M}_i$.

Using this in the previous expression, we have

$$\nabla_{\mathbf{x}_0} \mathcal{J}(\mathbf{x}_N) = \mathbf{M}_0^T(\mathbf{x}_0)\,\mathbf{M}_1^T(\mathbf{x}_1)\cdots\mathbf{M}_{N-1}^T(\mathbf{x}_{N-1})\,\nabla_{\mathbf{x}_N}\mathcal{J}(\mathbf{x}_N)$$

Define a variable λ_i that satisfies

$$\begin{aligned} \lambda_i &= \mathbf{M}_i^T(\mathbf{x}_i)\,\lambda_{i+1}, \quad i = N-1,\ldots,0 && (14) \\ \lambda_N &= \nabla_{\mathbf{x}_N}\mathcal{J}(\mathbf{x}_N) && (15) \end{aligned}$$

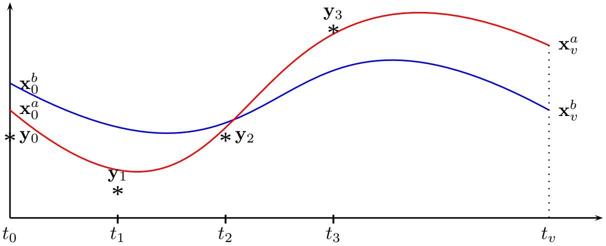
this is the discrete FOA model of (10) and is integrated backwards in time. As with the continuous case $\lambda_0 = \nabla_{\mathbf{x}_0}\mathcal{J}(\mathbf{x}_N)$

Discrete Computation of TLM and FOA

$$
\begin{tikzcd}
\text{TLM} & \mathbf{w} \arrow[r] & \boxed{\mathbf{M}_0\left(\mathbf{x}_0\right)} \arrow[r] & \mu_1 \arrow[r] & \boxed{\mathbf{M}_1\left(\mathbf{x}_1\right)} \arrow[r] & \mu_2 & \cdots & \mu_{N-1} \arrow[r] & \boxed{\mathbf{M}_{N-1}\left(\mathbf{x}_{N-1}\right)} \arrow[r] & \mu_N \\
\text{forward model} & \mathbf{x}_0 \arrow[r] \arrow[ur] \arrow[dr] & \boxed{\mathcal{M}_0\left(\mathbf{x}_0\right)} \arrow[r] & \mathbf{x}_1 \arrow[r] \arrow[ur] \arrow[dr] & \boxed{\mathcal{M}_1\left(\mathbf{x}_1\right)} \arrow[r] & \mathbf{x}_2 & \cdots & \mathbf{x}_{N-1} \arrow[r] \arrow[ur] \arrow[dr] & \boxed{\mathcal{M}_{N-1}\left(\mathbf{x}_{N-1}\right)} \arrow[r] & \mathbf{x}_N \arrow[d] \\
\text{FOA} & \lambda_0 & \boxed{\mathbf{M}_0^*\left(\mathbf{x}_0\right)} \arrow[l] & \lambda_1 \arrow[l] & \boxed{\mathbf{M}_1^*\left(\mathbf{x}_1\right)} \arrow[l] & \lambda_2 \arrow[l] & \cdots & \lambda_{N-1} & \boxed{\mathbf{M}_{N-1}^*\left(\mathbf{x}_{N-1}\right)} \arrow[l] & \lambda_N \arrow[l]
\end{tikzcd}
$$

Figure: Flow chart for the computation of the tangent linear model and the adjoint model.

Four Dimensional Variational Method



The four dimensional variational data assimilation (4D-Var) considers all observations in a given time window to compute an updated model solution. The methodology is to optimize a cost function $\mathcal{J}$ with respect to initial conditions, parameters, boundary conditions, etc.
The cost function then becomes

$$\mathcal{J}(\mathbf{x}) = \left(\mathbf{x} - \mathbf{x}^f\right)^T \left(\mathbf{P}^f\right)^{-1} \left(\mathbf{x} - \mathbf{x}^f\right) + \sum_{k=1}^{T} \left(\mathbf{y}_k^o - \mathbf{H}_k \mathbf{x}_k\right)^T \mathbf{R}_k^{-1} \left(\mathbf{y}_k^o - \mathbf{H}_k \mathbf{x}_k\right)$$

2-D Shallow Water Model

A global 2D shallow water (SW) model on a sphere is used for the numerical experiments.

- Model describes hydrodynamic flow on a sphere assuming vertical motion is much smaller than horizontal motion.
- Assume fluid depth is small compared with radius of the sphere (radius of Earth).
- Computations done on a $2.5^\circ \times 2.5^\circ$ grid with a time step $\Delta t = 450$s.
- $\mathbf{x}_0^t$: trajectory produced by SW integration with I.C. taken from ERA-40 for March 15 2002 at $06:00$h.
- TLM and FOA obtained with automatic differentiation (TAMC).
- 20 leading eigenpairs of $\mathbf{M}^{*\mathbf{E}}\mathbf{M}$ computed with ARPACK.

Compute sensitivity of the SW model with respect to initial conditions.

Gradient Fields for SW

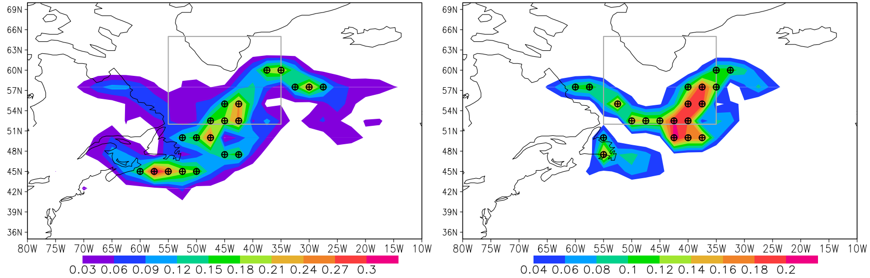


Figure: Gradient fields computed with a final time $t_N = t_0 = 24\text{h}$. Right figure: sensitivity field at $t_i = t_0$ left figure: sensitivity field at $t_i = t_0 + 6\text{h}$

Data Assimilation - 24h forecast error

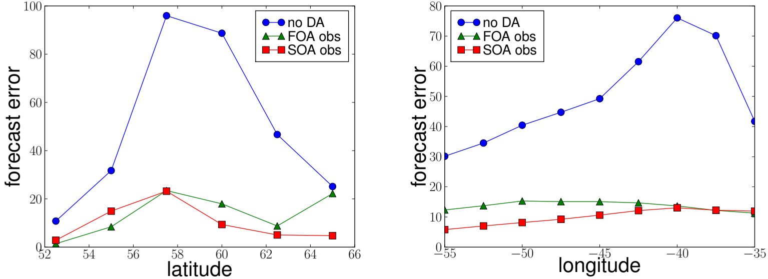


Figure: Longitudinal (left) and latitudinal (right) forecast error average over the target domain.

4D-Var data assimilation for SW with assimilation window $[0\mathrm{h}, 6\mathrm{h}]$.
Experiment with 20 adaptive observations at $t = 0$ and 6 hours.

Conclusions

- Adjoint models are extremely helpful for optimization problems
- Such optimization problems arise in data assimilation, where dimension of models are very high $(10^7–10^9)$
- Different form of adjoint computation can be performed, either discrete or continuous
- Automatic differentiation is a very useful tool for computing discrete adjoint models of complex code

Challenges ahead

- complex dynamical models with parameterizations are challenging for computing adjoint models
- significant up-front cost for developing and later maintaining adjoint models
- no actual reliability index associated with adjoint models
- models with incomplete physics $\rightarrow$ models error are challenging for adjoint information